What is claimed:

1. A method for allocating a memory block, comprising the steps of:

determining the size of a requested memory block;

allocating a memory block directly from an operating system if it is determined that the memory block is a predetermined large size;

allocating a memory block from an active memory super-block if it is determined that the memory block is a predetermined regular size;

allocating a memory block from a partial super-block if the step of allocating a memory block directly from the active memory block fails;

allocating a memory block from a new super-block if the step of allocating a memory block from the partial super block fails; and

returning the memory block in response to the request;

2. The method of claim 1, further comprising the step of allocating a memory block from the active super-block if a memory block fails to be allocated.

3. The method of claim 2, wherein the step of allocating a memory block from an active super-block further comprises the step of reading an active field of a memory block's heap header, wherein if a pointer value read from the active field is null then the allocation of the memory block fails, and if the pointer value is not null then a thread checks if a value of credits read from the active field is zero or more.

4. The method of claim 3, wherein if the value of credits is zero and the anchor field holds the same active field value as previously read, then the thread updates the active field to a null value, and if the value of credits is one or more and the anchor field holds the same active field value as previously read then the thread tries to atomically update the active field by decrementing the credit value.

5. The method of claim 4, wherein if the updating of the active field fails, then the thread reads a fresh value from the anchor field and if the updating of the active field succeeds, then the thread reads the anchor field descriptor values corresponding to block availability, count and state of a super-block.

6. The method of claim 5, wherein if the value of credits was more than zero and the anchor field descriptor of the super-block holds the same availability value as previously read, then the thread atomically updates the anchor field by setting the available value to an index of a next available memory block.

7. The method of claim 6, wherein if the value of credits equals zero and the count value of the super-block equals zero and the anchor field descriptor of the super-block holds the same count value as previously read, then the thread atomically updates the anchor field of the super-block by setting the state value to FULL and if the value of credits is not equal to zero and the count value of the super-block is greater than zero and the anchor field descriptor of the super-block holds the same count value as previously read, then the thread atomically updates the anchor field of the super-block by setting the availability value to the index of the next available memory block and decrementing the count value.

8. The method of claim 7, wherein if the updating of the active field of the super-block fails, then the thread reads a fresh value from the anchor field of the super-block.

9. The method of claim 8, wherein if the thread has decremented the count value of the anchor field of the super-block, then the thread updates the value of credits of the active field of a memory block's heap header.

10. The method of claim 9, wherein the step of allocating a memory block from a partial super-block further comprises the step of locating a non-empty super-block from a list of partial super-blocks designated within a predetermined size class, wherein if an attempt to locate a non-empty super-block returns a null value, then the step of locating a non-empty super-block fails, and if a non-empty super-block is located, then the thread reads the anchor field of the super-block descriptor for values corresponding to for values corresponding to availability, count and state.

11. The method of claim 10, wherein if the count value is equal to one and the anchor field descriptor of the super-block holds the same value as previously read, then the thread atomically updates the anchor field by setting the state value to FULL, and if the count value is greater than one and the anchor field descriptor of the super-block holds the same value as previously read, then the thread atomically updates the anchor field by setting the availability value to the index of the next available block and decrement the credits of the count value and further, if the updating of the anchor field of the super-block fails, then the thread reads a fresh value from the anchor field of the super-block.

12. The method of claim 11, wherein if the thread has decremented the count value of the anchor field of the super-block, then the thread updates the value of credits of the active field of a memory block's heap header.

13. The method of claim 12, wherein the step of allocating a memory block from a new super-block further comprises the step of initializing the value fields of the new super-block and organizing the new super-block into memory blocks of a predetermined size and atomically updating the anchor field descriptor of a memory by installing the address of the anchor field descriptor and installing an initial credit value of the new super-block in the heap header of the memory block.

14. The method of claim 13, wherein if the updating of the heap header fails, then the anchor field descriptor is retired and the super-block is freed.

15. The method of claim 14, wherein the step of updating the credit value of a heap header comprises the step of the thread atomically updating the anchor field of a heap header by setting the active pointer to the address of an associated super-block descriptor and decrementing the credit value by one of the credit value decremented earlier from the count value of the anchor field of the super-block descriptor, if the active field is determined to hold a null value.

16. The method of claim 15, wherein if the atomic updating of the anchor field of the heap header fails, then the thread atomically updates the anchor field by adding the credit value to the count value and setting the state value to PARTIAL.

17. The method of claim 16, further comprising the step of inserting the super-block into a list of super-blocks that are associated with a predetermined size class.

18. A method for deallocating a previously allocated memory block, comprising the steps of:

determining the size of a memory block, wherein the memory block is returned to the operating system if it is determined that the block is a large block;

reading an anchor field descriptor of an associated memory super-block in order to acquire an availability descriptor, count descriptor and state descriptor value of the memory super-block;

determining if the memory super-block is full or not full; and

atomically updating the anchor field descriptor of the associated memory super-block.

19. The method of claim 18, wherein if the memory super-block is determined to be full, then a thread tries to atomically update the anchor field of the associated super-block descriptor.

20. The method of claim 19, wherein if the memory super-block is determined to be not full, then the thread checks if all of the other blocks in the super-block are available and that the memory super-block is found to be not active , wherein if the memory super-block is not active and the anchor field holds the same state value as previously read, then the thread tries to atomically update the anchor field by setting the anchor field state to empty.

21. The method of claim 20, wherein if the memory super-block is determined to be not full, then the thread checks if all of the other blocks in the super-block are available and that the memory super-block is not active, wherein if the memory super-block is found to be active and the anchor field holds the same state value as previously read, then the thread tries to atomically update the anchor field values by setting the availability descriptor to an index of the deallocated memory block and incrementing the value of the count descriptor.

22. The method of claim 21, wherein if the step of atomically updating of the anchor field descriptor of the associated super-block fails because the value of the anchor field differs from the value previously read from the anchor field value, then the thread reads and acquires fresh anchor field values.

23. The method of claim 22, wherein if the read value of the state descriptor of the anchor field was full, the thread must insert the memory super-block descriptor into a list of partially full memory super-blocks.

24. The method of claim 23, wherein if the new state descriptor of the anchor field is empty, the memory super-block is returned to the operating system.

25. The method of claim 23, wherein if the new state descriptor of the anchor field is empty, the memory super-block is placed on a list of empty memory super-blocks.

26. A computer program product that includes a computer readable medium useable by a processor, the medium having stored thereon a sequence of instructions which, when executed by the processor, causes the processor to allocate a memory block, wherein the computer program product executes the steps of:

determining the size of a requested memory block;

allocating a memory block directly from an operating system if it is determined that the memory block is a predetermined large size;

allocating a memory block from an active memory super-block if it is determined that the memory block is a predetermined regular size;

allocating a memory block from a partial super-block if the step of allocating a memory block directly from the active memory block fails;

allocating a memory block from a new super-block if the step of allocating a memory block from the partial super block fails; and

returning the memory block in response to the request.

27. The computer program product of claim 26, further comprising the step of allocating a memory block from the active super-block if a memory block fails to be allocated.

28. The computer program product of claim 27, wherein the step of allocating a memory block from an active super-block further comprises the step of reading an active field of a memory block's heap header, wherein if a pointer value read from the active field is null, then the

allocation of the memory block fails and if the pointer value is not null, then a thread checks if a value of credits read from the active field is zero or more.

29. The computer program product of claim 28, wherein if the value of credits is zero and the anchor field holds the same active field value as previously read, then the thread updates the active field to a null value and if the value of credits is one or more and the anchor field holds the same active field value as previously read, then the thread tries to atomically update the active field by decrementing the credit value.

30. The computer program product of claim 29, wherein if the updating of the active field fails, then the thread reads a fresh value from the anchor field and if the updating of the active field succeeds, then the thread reads the anchor field descriptor values corresponding to block availability, count and state of a super-block.

31. The computer program product of claim 30, wherein if the value of credits was more than zero and the anchor field descriptor of the super-block holds the same availability value as previously read, then the thread, then the thread atomically updates the anchor field by setting the available value to an index of a next available memory block.

32. The computer program product of claim 31, wherein if the value of credits equals zero and the count value of the super-block equals zero and the anchor field descriptor of the super-block holds the same count value as previously read, then the thread atomically updates the anchor field of the super-block by setting the state value to FULL and if the value of credits is not equal to zero and the count value of the super-block is greater than zero and the anchor field descriptor of the super-block holds the same count value as previously read, then the thread atomically updates the anchor field of the super-block by setting the availability value to the index of the next available memory block and decrementing the count value.

33. The computer program product of claim 32, wherein if the updating of the active field of the super-block fails, then the thread reads a fresh value from the anchor field of the super-block.

34. The computer program product of claim 33, wherein if the thread has decremented the count value of the anchor field of the super-block, then the thread updates the value of credits of the active field of a memory block's heap header.

35. The computer program product of claim 34, wherein the step of allocating a memory block from a partial super-block further comprises the step of locating a non-empty super-block from a list of partial super-blocks designated within a predetermined size class, wherein if an attempt to locate a non-empty super-block returns a null value, then the step of locating a non-empty super-block fails, and if a non-empty super-block is located, then the thread reads the anchor field of the super-block descriptor for values corresponding to for values corresponding to availability, count and state.

36. The computer program product of claim 35, wherein if the count value is equal to one and the anchor field descriptor of the super-block holds the same count value as previously read, then the thread atomically updates the anchor field by setting the state value to FULL, and if the count value is greater than one and the anchor field descriptor of the super-block holds the same count value as previously read, then the thread atomically updates the anchor field by setting the availability value to the index of the next available block and decrement the credits of the count value and further, if the updating of the anchor field of the super-block fails, then the thread reads a fresh value from the anchor field of the super-block.

37. The computer program product of claim 36, wherein if the thread has decremented the count value of the anchor field of the super-block, then the thread updates the value of credits of the active field of a memory block's heap header.

38. The computer program product of claim 37, wherein the step of allocating a memory block from a new super-block further comprises the step of initializing the value fields of the new super-block and organizing the new super-block into memory blocks of a predetermined size and atomically updating the anchor field descriptor of a memory by installing the address of the

anchor field descriptor and installing an initial credit value of the new super-block in the heap header of the memory block.

39. The computer program product of claim 38, wherein if the updating of the heap header fails, then the anchor field descriptor is retired and the super-block is freed.

40. The computer program product of claim 39, wherein the step of updating the credit value of a heap header comprises the step of the thread atomically updating the anchor field of a heap header by setting the active pointer to the address of an associated super-block descriptor and decrementing the credit value by one of the credit value decremented earlier from the count value of the anchor field of the super-block descriptor, if the active field is determined to hold a null value.

41. The computer program product of claim 40, wherein if the atomic updating of the anchor field of the heap header fails, then the thread atomically updates the anchor field by adding the credit value to the count value and setting the state value to PARTIAL.

42. The computer program product of claim 41, further comprising the step of inserting the super-block into a list of super-blocks that are associated with a predetermined size class.

43. A computer program product that includes a computer readable medium useable by a processor, the medium having stored thereon a sequence of instructions which, when executed by the processor, causes the processor to deallocate a memory block, wherein the computer program product executes the steps of:

> determining the size of a memory block, wherein the memory block is returned to the operating system if it is determined that the block is a large block;

reading an anchor field descriptor of an associated memory super-block in order to acquire an availability descriptor, count descriptor and state descriptor value of the memory super-block;

determining if the memory super-block is full or not full; and

atomically updating the anchor field descriptor of the associated memory super-block.

44. The computer program product of claim 43, wherein if the memory super-block is determined to be full, then a thread tries to atomically update the anchor field of the associated super-block descriptor.

45. The computer program product of claim 44, wherein if the memory super-block is determined to be not full, then the thread checks if all of the other blocks in the super-block are available and that the memory super-block is found to be not active, wherein if the memory super-block is not active and the anchor field holds the same state value as previously read, then the thread tries to atomically update the anchor field by setting the anchor field state to empty.

46. The computer program product of claim 45, wherein if the memory super-block is determined to be not full, then the thread checks if all of the other blocks in the super-block are available and that the memory super-block is not active, wherein if the memory super-block is found to be active and the anchor field holds the same state value as previously read, then the thread tries to atomically update the anchor field values by setting the availability descriptor to an index of the deallocated memory block and incrementing the value of the count descriptor.

47. The computer program product of claim 46, wherein if the step of atomically updating of the anchor field descriptor of the associated super-block fails because the value of the anchor field differs from the value previously read from the anchor field value, then the thread reads and acquires fresh anchor field values.

48. The computer program product of claim 47, wherein if the read value of the state descriptor of the anchor field was full, the thread must insert the memory super-block descriptor into a list of partially full memory super-blocks.

49. The computer program product of claim 48, wherein if the new state descriptor of the anchor field is empty, the memory super-block is returned to the operating system.

50. The computer program product of claim 48, wherein if the new state descriptor of the anchor field is empty, the memory super-block is placed on a list of empty memory super-blocks.

51. A computer system for allocating a memory block, comprising:

a memory block size determining means for determining the size of a requested memory block;

a memory block allocating means for allocating a memory block directly from an operating system if it is determined that the memory block is a predetermined large size and for allocating a memory block from an active memory super-block if it is determined that the memory block is a predetermined regular size, wherein the memory block allocation means allocates a memory block from a partial super-block if the step of allocating a memory block directly from the active memory block fails, and allocating a memory block from a new super-block if the step of allocating a memory block from the partial super block fails.

52. A computer system for deallocating a previously allocated memory block, comprising:

a memory block size determination means for determining the size of a memory block, wherein the memory block is returned to the operating system if it is determined that the memory block is a large block;

a means to read the anchor field descriptor of an associated memory super-block in order to acquire an availability descriptor, count descriptor and state descriptor value of the memory super-block;

a means to determine if the memory super-block is full or not full; and

a means to atomically update the anchor field descriptor of the associated memory super-block.